Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	101	nest$3 adj class	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:05
L2	3	nest$3 adj class with template	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:19
L3	10577	template with (input or source)	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:19
L4	1836	template with (input or source) with (output or target)	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:20
L6	191	template with (input or source) with (output or target) with combin$5	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:21
L7	0	template with class with (input or source) with (output or target) with combin$5	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:21
L8	35	template with class with (input or source) with (output or target)	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:21
L9	66	template with class with combin$5	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:21
S2	112	inner adj class	US-PGPUB; USPAT	OR	ON	2005/09/30 17:25
S3	10	inner adj class same template	US-PGPUB; USPAT	OR	ON	2005/09/30 17:27
S4	4	nested adj class same template	US-PGPUB; USPAT	OR	ON	2005/09/30 17:27
S5	15	("5172326" | "5287449" | "5790855" | "5835577" | "5864700" | "6032152" | "6063128" | "6106569" | "6151703" | "6173276" | "6199195" | "6268853" | "6300948" | "6405368" | "6785673").PN.	US-PGPUB; USPAT; USOCR	OR	ON	2005/10/01 09:05

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: (●) The ACM Digital Library ( ) The Guide

template and inner class SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **template** and **inner** **class**

Found **10,168** of **161,645**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Session 2 (short papers): system and practical issues: Collaboration-based evolvable software implementations: Java and Hyper/J vs. C++-templates composition
Nguyen Truong Thang, Takuya Katayama
May 2002 **Proceedings of the International Workshop on Principles of Software Evolution**
Full text available: pdf(424.71 KB) Additional Information: full citation, abstract, references

Collaboration-based design is a methodology for de-coupling application aspects in the object-oriented world. Because collaborations are relatively orthogonal to each other, a collaboration can be implemented independently and reused in different context without any major change. Mixins and mixin layers are techniques to transform in a fairly straightforward manner the collaboration-based designs to the implementations. Subject-oriented programming (SOP) [3] and multi-dimensional separation of c ...

2 Mixin layers: an object-oriented implementation technique for refinements and collaboration-based designs
Yannis Smaragdakis, Don Batory
April 2002 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 11 Issue 2
Full text available: pdf(510.43 KB) Additional Information: full citation, abstract, references, citings, index terms

A "refinement" is a functionality addition to a software project that can affect multiple dispersed implementation entities (functions, classes, etc.). In this paper, we examine large-scale refinements in terms of a fundamental object-oriented technique called collaboration-based design. We explain how collaborations can be expressed in existing programming languages or can be supported with new language constructs (which we have implemented as extensions to the Java language). We present a spec ...

Keywords: Collaboration-based design, component-based software, product-line architectures

3 A modest proposal: C++ resyntaxed
Ben Werther, Damian Conway
November 1996 **ACM SIGPLAN Notices**, Volume 31 Issue 11
Full text available: pdf(745.78 KB) Additional Information: full citation, abstract, index terms

We describe an alternative syntactic binding for C++. This new binding includes a completely redesigned declaration/definition syntax for types, functions and objects, a simplified template syntax, and changes to several problematic operators and control structures. The resulting syntax is LALR(1) parsable and provides better consistency in the specification of similar constructs, better syntactic differentiation of dissimilar constructs, and greater overall readability of code.

4 Query optimization II: An architecture for query optimization
Arnon Rosenthal, David Reiner
June 1982 **Proceedings of the 1982 ACM SIGMOD international conference on Management of data**
Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings

We describe an optimizer for relational queries to databases stored as flat files and Codasyl networks. We include sophisticated manipulations on a broad range of direct access structures (DAS's). To achieve this with minimum additional code, we allow operations like sort, scan, and join to apply to DAS's, and categorize indexes and other DAS's in terms of the operations which can be performed on them. Our storage model, based on indivisible units of access and a small set of associated physical ...

5 A template-based approach to XML parsing in C++
John Dubchak
June 2003 **Linux Journal**, Volume 2003 Issue 110
Full text available: html(13.51 KB) Additional Information: full citation, abstract, index terms

Add XML support to your project with the Apache Software Foundation's Xerces parser and some C++ code.

6 Flattening and parallelizing irregular, recurrent loop nests
Anwar M. Ghuloum, Allan L. Fisher
August 1995 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 30 Issue 8
Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references, citings, index terms

Irregular loop nests in which the loop bounds are determined dynamically by indexed arrays are difficult to compile into expressive parallel constructs, such as segmented scans and reductions. In this paper, we describe a suite of transformations to automatically parallelize such irregular loop nests, even in the presence of recurrences. We describe a simple, general loop flattening transformation, along with new optimizations which make it a viable compiler transformation. A robust recurre ...

7 Reviewed papers: A framework for applet animations with controls
Chenglie Hu
December 2003 **ACM SIGCSE Bulletin**, Volume 35 Issue 4
Full text available: pdf(316.05 KB) Additional Information: full citation, abstract, references

A generic Java™ applet animation with controls is presented in this paper. This framework implementation promotes software reusability in terms of applying modularity and factorization using inheritance, delegation, and template classes with cohesive methods. It can be used either as an example or a project in a typical object-oriented design and analysis class.

8 Super and inner: together at last!
David S. Goldberg, Robert Bruce Findler, Matthew Flatt
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN**

Conference on Object-oriented programming, systems, languages, and applications, Volume 39 Issue 10
Full text available: pdf(181.07 KB) Additional Information: full citation, abstract, references, index terms

In an object-oriented language, a derived class may declare a method with the same signature as a method in the base class. The meaning of the re-declaration depends on the language. Most commonly, the new declaration overrides the base declaration, perhaps completely replacing it, or perhaps using <b>super</b> to invoke the old implementation. Another possibility is that the base class always controls the method implementation, and the new declaration merely augments the method in th ...

Keywords: augment, inheritance, inner, override, super

9 Interfaces as specifications in the MIDAS user interface development systems
Regina H. B. Cabral, Ivan M. Campos, Donald D. Cowan, Carlos J. P. Lucena
April 1990 **ACM SIGSOFT Software Engineering Notes**, Volume 15 Issue 2
Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, index terms

This paper describes an evolving User Interface Development System called MIDAS (for Merging Interface Development with Application Specification) which allows interface/systems designers to develop an application-specific user interface interactively, in a prototyping-oriented environment, while refining the specification of the intended application itself. The interface/systems designer receives expert advice on both interface and application software design principles, emerging from MIDAS' kn ...

Keywords: direct manipulation interfaces, expert assistance, knowledge bases, lifecycle model, object-oriented design and development, prototyping, specifications, user interface development systems, user interface management systems, user interfaces, user models

10 Teaching object-orientation with the Object Visualization and Annotation Language (OVAL)
Mirko Raner
July 2000 **ACM SIGCSE Bulletin , Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education**, Volume 32 Issue 3
Full text available: pdf(484.38 KB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented techniques and technologies are omnipresent in all branches of modern software development and systems design. Still today there is an enormous demand for training in the area of object-oriented analysis, design and programming.Several languages and notations have been developed for the visual presentation of object-oriented ideas and designs (eg, the Booch method [1], OMT [3] or the emerging standard UML [4]). Such languages or notations are an excellent means of communication a ...

11 Coordination models, languages and applications (CM): LighTS: a lightweight, customizable tuple space supporting context-aware applications
Gian Pietro Picco, Davide Balzarotti, Paolo Costa
March 2005 **Proceedings of the 2005 ACM symposium on Applied computing**
Full text available: pdf(200.86 KB) Additional Information: full citation, abstract, references

The tuple space model inspired by Linda has recently been rediscovered by distributed middleware. Moreover, some researchers also applied it in the challenging scenarios involving mobility and more specifically context-aware computing. Context information can be stored in the tuple space, and queried like any other data.Nevertheless, it turns out that conventional tuple space implementations fall short of expectations in this new domain. On one hand, many of the available systems provide a wealt ...

12 A truly generative semantics-directed compiler generator
Harald Ganzinger, Robert Giegerich, Ulrich Möncke, Reinhard Wilhelm
June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6
Full text available: pdf(918.86 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes semantic processing in the compiler generating system MUG2. MUG2 accepts high-level descriptions of the semantics of a programming language including full runtime semantics, data flow analysis, and optimizing transformations. This distinguishes MUG2 from systems such as YACC [Joh75], HLP [HLP78], PQCC [PQC79], or its own former version [GRW77] with respect to expressive power and convenience. In this respect, MUG2 comes close to semantics-directed systems such as [Mos76 ...

13 Programming languages (PL): Type-safe covariance in C++
Vitaly Surazhsky, Joseph (Yossi) Gil
March 2004 **Proceedings of the 2004 ACM symposium on Applied computing**
Full text available: pdf(194.42 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a programming technique for implementing type safe covariance in C++. In a sense, we implement most of Bruce's *matching* approach to the covariance dilemma in C++. The appeal in our approach is that it relies on existing mechanisms, specifically templates, and does not require any modification to the existing language. The practical value of the technique was demonstrated in its successful incorporation in a large software body. We identify the ingredients of a programming langu ...

Keywords: C++, covariance, templates, type safety

14 Efficient crosstalk noise modeling using aggressor and tree reductions
Li Ding, David Blaauw, Pinaki Mazumder
November 2002 **Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design**
Full text available: pdf(139.91 KB) Additional Information: full citation, abstract, references, index terms

This paper describes a fast method to estimate crosstalk noise in the presence of multiple aggressor nets for use in physical design automation tools. Since noise estimation is often part of the innerloop of optimization algorithms, very efficient closed-form solutions are needed. Previous approaches have typically used simple lumped 3--4 node circuit templates. One aggressor net is modeled at a time assuming that the coupling capacitances to all quiet aggressor nets are grounded. They also mode ...

15 Change management: An infrastructure for development of object-oriented, multi-level configuration management services
Tien N. Nguyen, Ethan V. Munson, John T. Boyland, Cheng Thao
May 2005 **Proceedings of the 27th international conference on Software engineering**
Full text available: pdf(418.74 KB) Additional Information: full citation, abstract, references, index terms

In an integrated development environment, the ability to manage the evolution of a software system in terms of logical abstractions, compositions, and their interrelations is crucial to successful software development. This paper presents a novel framework and infrastructure, *Molhado*, upon which to build *object-oriented* software configuration management (SCM) services in a SCM-centered integrated development environment. Key contributions of this paper include a *product versioni ...*

***Keywords**: software configuration management, version control*

16 Using testing and JUnit across the curriculum
Michael Wick, Daniel Stevenson, Paul Wagner
February 2005 **ACM SIGCSE Bulletin , Proceedings of the 36th SIGCSE technical symposium on Computer science education**, Volume 37 Issue 1
Full text available: pdf(219.93 KB) Additional Information: full citation, abstract, references, index terms

While the usage of unit-testing frameworks such as JUnit has greatly increased over the last several years, it is not immediately apparent to students and instructors how to best use tools like JUnit and how to integrate testing across a computer science curriculum. We have worked over the last four semesters to infuse testing and JUnit across our curriculum, building from having students use JUnit to having them write their own test cases to building larger integration and use case testing syst ...

Keywords: JUnit, testing, unit testing, unit testing frameworks

17 On the role of language constructs for framework design
Görel Hedin, Jørgen Lindskov Knudsen
March 2000 **ACM Computing Surveys (CSUR)**
Full text available: pdf(37.97 KB) Additional Information: full citation, references, index terms

18 MixJuice (poster session): an object-oriented language with simple and powerful module mechanism
Yuuji Ichisugi
January 2000 **Addendum to the 2000 proceedings of the conference on Object-oriented programming, systems, languages, and applications (Addendum)**
Full text available: pdf(64.48 KB) Additional Information: full citation, abstract, references, index terms

This paper describes an object-oriented language called *MixJuice*, which uses the differences of class hierarchies as a unit of information hiding and reuse. In this language, classes — templates of objects — and modules — units of information hiding and reuse — are completely orthogonal, thus supporting *separation of concerns*. MixJuice is basically based on Java. However, its module mechanism is simpler and more powerful than that of Java. The MixJuice pr ...

19 Facilitating abstraction and reuse with ExpectTK
Navid Sabbaghi
February 1996 **Crossroads**, Volume 2 Issue 3
Full text available: html(36.36 KB) Additional Information: full citation, index terms

20 Dependency diagrams
Mark Ray
February 1996 **Crossroads**, Volume 2 Issue 3
Full text available: html(36.36 KB) Additional Information: full citation, index terms

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

"anonymous class" +java SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used anonymous class java Found **8,442** of **161,645**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown

Relevance scale

1 Chianti: a tool for change impact analysis of java programs
Xiaoxia Ren, Fenil Shah, Frank Tip, Barbara G. Ryder, Ophelia Chesley
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(465.37 KB) Additional Information: full citation, abstract, references, index terms

This paper reports on the design and implementation of Chianti, a change impact analysis tool for Java that is implemented in the context of the Eclipse environment. Chianti analyzes two versions of an application and decomposes their difference into a set of atomic changes. Change impact is then reported in terms of affected (regression or unit) tests whose execution behavior may have been modified by the applied changes. For each affected test, Chianti also determines a set of affecting cha ...

Keywords: analysis of object-oriented programs, change impact analysis, regression test, unit test

2 On objects and events
Patrick Th. Eugster, Rachid Guerraoui, Christian Heide Damm
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications**, Volume 36 Issue 11
Full text available: pdf(308.58 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents linguistic primitives for publish/subscribe programming using events and objects. We integrate our primitives into a strongly typed object-oriented language through four mechnisms: (1) serialization, (2) multiple subtyping, (3)closures, and (4) deferred code evaluation. We illustrate our primitives through Java, showing how we have overcome its respective lacks. A precompiler transforms statements based on our publish/subscribe primitives into calls to specifically generated ...

3 Scalable extensibility via nested inheritance
Nathaniel Nystrom, Stephen Chong, Andrew C. Myers
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10

Full text available: pdf(196.74 KB) Additional Information: full citation, abstract, references, index terms

Inheritance is a useful mechanism for factoring and reusing code. However, it has limitations for building extensible systems. We describe <i>nested inheritance</i>, a mechanism that addresses some of the limitations of ordinary inheritance and other code reuse mechanisms. Using our experience with an extensible compiler framework, we show how nested inheritance can be used to construct highly extensible software frameworks. The essential aspects of nested inheritance are formalized i ...

Keywords: inheritance, nested classes, object-oriented programming languages, virtual classes

4 Extensible algebraic datatypes with defaults
Matthias Zenger, Martin Odersky
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming**, Volume 36 Issue 10
Full text available: pdf(259.40 KB) Additional Information: full citation, abstract, references, citings, index terms

A major problem for writing extensible software arises when recursively defined datatypes and operations on these types have to be extended simultaneously without modifying existing code. This paper introduces Extensible Algebraic Datatypes with defaults, which promote a simple programming pattern to solve this well-known problem. We show that it is possible to encode extensible algebraic datatypes in an object-oriented language, using a new design pattern for extensible visitors. Extensible alg ...

5 Technical correspondence: Closures for statically-typed object-oriented languages
José de Oliveira Guimarães
August 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 8
Full text available: pdf(117.69 KB) Additional Information: full citation, abstract, references

Closures increase considerably the level of a language by mixing access to local variables with remote execution of a set of locally-defined statements. However, to date closures have not been added to statically-typed languages because it is difficult to type them and runtime errors occur if local variables that no longer exist are accessed. We proposed a limited but quite general kind of closure for statically-typed object-oriented languages. They can be used in most situations normal closu ...

Keywords: Smalltalk blocks, closure, green, object-oriented languages

6 Converting java programs to use generic libraries
Alan Donovan, Adam Kiežun, Matthew S. Tschantz, Michael D. Ernst
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, index terms

Java 1.5 will include a type system (called JSR-14) that supports <i>parametric polymorphism</i>, or <i>generic</i> classes. This will bring many benefits to Java programmers, not least because current Java practice makes heavy use of logically-generic classes, including container classes.

Translation of Java source code into semantically equivalent JSR-14 source code requires two steps: parameterization (adding type parameters to class definitions) and instantiation (a ...

Keywords: JSR-14, Java 1.5, Java 5, generic types, instantiation types, parameterized types, parametric polymorphism, raw types, type inference

7 Making the future safe for the past: adding genericity to the Java programming language
Gilad Bracha, Martin Odersky, David Stoutamire, Philip Wadler
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 33 Issue 10
Full text available: pdf(1.91 MB) Additional Information: full citation, abstract, references, citings, index terms

We present GJ, a design that extends the Java programming language with generic types and methods. These are both explained and implemented by translation into the unextended language. The translation closely mimics the way generics are emulated by programmers: it erases all type parameters, maps type variables to their bounds, and inserts casts where needed. Some subtleties of the translation are caused by the handling of overriding.GJ increases expressiveness and safety: code utilizing generic ...

8 Technical papers: software maintenance: Concern graphs: finding and describing concerns using structural program dependencies
Martin P. Robillard, Gail C. Murphy
May 2002 **Proceedings of the 24th International Conference on Software Engineering**
Full text available: pdf(1.35 MB) Additional Information: full citation, abstract, references, citings, index terms

Many maintenance tasks address concerns, or features, that are not well modularized in the source code comprising a system. Existing approaches available to help software developers locate and manage scattered concerns use a representation based on lines of source code, complicating the analysis of the concerns. In this paper, we introduce the Concern Graph representation that abstracts the implementation details of a concern and makes explicit the relationships between different parts of the co ...

9 Performance and Analysis: WebGraph: things you thought you could not do with Java™
Paolo Boldi, Sebastiano Vigna
June 2004 **Proceedings of the 3rd international symposium on Principles and practice of programming in Java PPPJ '04**
Full text available: pdf(180.33 KB) Additional Information: full citation, abstract, references

Studying web graphs is often difficult due to their large size. The WebGraph framework is a suite of codes, algorithms and tools that make it easy to manipulate large web graphs, and to store them in a limited space, by exploiting the inner redundancies of the web. WebGraph is based on sophisticated bitwise compression techniques, and functional-style lazy constructions. Common wisdom would say that the most unlikely language to implement such a framework is Java. We are going to tell you the re ...

10 Compatible genericity with run-time types for the Java programming language
Robert Cartwright, Guy L. Steele
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 33 Issue 10
Full text available: pdf(1.97 MB) Additional Information: full citation, abstract, references, citings, index terms

The most serious impediment to writing substantial programs in the Java&trade; programming language is the lack of a *gentricity* mechanism for abstracting classes and methods with respect to type. During the past two years, several research groups have developed Java extensions that support various forms of genericity, but none has succeeded in accommodating general type parameterization (akin to Java arrays) while retaining compatibility with the existing. Java Virtual Machine. In thi ...

11 Language-specific make technology for the Java programming language
Mikhail Dmitriev
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11
Full text available: pdf(238.19 KB) Additional Information: full citation, abstract, references, citings, index terms

Keeping the code of a Java application consistent (code is consistent if all of the project classes can be recompiled together without errors) prevents late linking errors, and thus may significantly improve development turnaround time. In this paper we describe a make technology for the Java programming language, that is based on smart dependency checking, guarantees consistency of the project code, and at the same time reduces the number of source code recompilations to the minimum. After proj ...

Keywords: Java, dependency checking, development turnaround time, make

12 A practical type system and language for reference immutability
Adrian Birka, Michael D. Ernst
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(171.73 KB) Additional Information: full citation, abstract, references, index terms

This paper describes a type system that is capable of expressing and enforcing immutability constraints. The specific constraint expressed is that the abstract state of the object to which an immutable reference refers cannot be modified using that reference. The abstract state is (part of) the transitively reachable state: that is, the state of the object and all state reachable from it by following references. The type system permits explicitly excluding fields or objects from the abstract ...

Keywords: Java, Javari, const, immutability, mutable, readonly, type system, verification

13 A selective, just-in-time aspect weaver
Yoshiki Sato, Shigeru Chiba, Michiaki Tatsubori
September 2003 **Proceedings of the second international conference on Generative programming and component engineering GPCE '03**
Full text available: pdf(256.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic AOP (Aspect-Oriented Programming) is receiving growing interests in both the academia and the industry. Since it allows weaving aspects with a program at runtime, it is useful for rapid prototyping and adaptive software. However, the previous implementations of dynamic AOP systems suffered from serious performance penalties. This paper presents our new efficient dynamic AOP system in Java for addressing the underlying problem. This system called Wool is a hybrid of two approaches. When a ...

14 Optimising aspectJ

Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, Julian Tibble
June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6
Full text available: pdf(194.20 KB) Additional Information: full citation, abstract, references, index terms

AspectJ, an aspect-oriented extension of Java, is becoming increasingly popular. However, not much work has been directed at optimising compilers for AspectJ. Optimising AOP languages provides many new and interesting challenges for compiler writers, and this paper identifies and addresses three such challenges.First, compiling *around* advice efficiently is particularly challenging. We provide a new code generation strategy for *around* advice, which (unlike previous implementations) ...

Keywords: around advice, aspect-oriented programming language, aspectJ, cflow pointcut, optimization

15 Semantic interfaces and OWL tools: Parsing owl dl: trees or triples?
Sean K. Bechhofer, Jeremy J. Carroll
May 2004 **Proceedings of the 13th international conference on World Wide Web**
Full text available: pdf(156.25 KB) Additional Information: full citation, abstract, references, citings, index terms

The Web Ontology Language (OWL) defines three classes of documents: Lite, DL, and Full. All RDF/XML documents are OWL Full documents, some OWL Full documents are also OWL DL documents, and some OWL DL documents are also OWL Lite documents. This paper discusses *parsing* and *species recognition* -- that is the process of determining whether a given document falls into the OWL Lite, DL or Full class. Wedescribe two alternative approaches to this task, one based on abstract syntax trees, ...

Keywords: owl, parsing, rdf, semantic web

16 Advanced control flow in Java card programming
Peng Li, Steve Zdancewic
June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems**, Volume 39 Issue 7
Full text available: pdf(205.46 KB) Additional Information: full citation, abstract, references, index terms

Java Card technology simplifies the development of smart card applications by providing a high-level programming language similar to Java. However, the master-slave programming model used in current Java Card platform creates control flow difficulties when writing complex card programs, making it inconvenient, tedious, and error-prone to implement Java Card applications. This paper examines these drawbacks of the master-slave model and proposes a concurrent thread model for developing future Jav ...

Keywords: CPS, Java card, continuation, control flow, smart card, trampolined style

17 Technical correspondence: Java RMI, RMI tunneling and Web services comparison and performance analysis
Matjaz B. Juric, Bostjan Kezmah, Marjan Hericko, Ivan Rozman, Ivan Vezocnik
May 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 5
Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references

This article compares different approaches for developing Java distributed applications which have to communicate through firewalls and proxies, including RMI over open ports, HTTP-to-port, HTTP-to-CGI, HTTP-to-servlet tunneling and web services. A functional comparison of approaches has been done, as well as a detailed performance analysis with overhead analysis and identification of optimizations. Therefore the paper contributes to the overall understanding of different approaches for developi ...

Keywords: RMI, SOAP, performance, tunneling, web services

18 Jedd: a BDD-based relational extension of Java
Ondřej Lhoták, Laurie Hendren
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(137.26 KB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we present Jedd, a language extension to Java that supports a convenient way of programming with Binary Decision Diagrams (BDDs). The Jedd language abstracts BDDs as database-style relations and operations on relations, and provides static type rules to ensure that relational operations are used correctly.The paper provides a description of the Jedd language and reports on the design and implementation of the Jedd translator and associated runtime system. Of particular interest is ...

Keywords: Java, binary decision diagrams, boolean formula satisfiability, language design, program analysis, relations

19 Theory & human computer interaction & programming languages: Adding polymorphic tuples and lists to the syntax of Java
Nattawut Sridranop, Ryan Stansifer
April 2004 **Proceedings of the 42nd annual Southeast regional conference**
Full text available: pdf(319.96 KB) Additional Information: full citation, abstract, references, index terms

Java does not provide built-in methods for constructing and accessing tuples and lists. In other programming languages, especially functional languages, these data structures are an integral part of the language. In fact, programming without tuples and lists is inconceivable in these languages.This paper will discuss a method for adding these capabilities to Java. We propose a syntax for tuples and lists that fits in with the existing Java syntax. We define a semantics for the new constructs bas ...

Keywords: Java, list, tuple

20 JR: Flexible distributed programming in an extended Java
Aaron W. Keen, Tingjian Ge, Justin T. Maris, Ronald A. Olsson
May 2004 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 26 Issue 3
Full text available: pdf(167.00 KB) Additional Information: full citation, abstract, references, index terms

Java provides a clean object-oriented programming model and allows for inherently system-independent programs. Unfortunately, Java has a limited concurrency model, providing only threads and remote method invocation (RMI).The JR programming language extends Java to provide a rich concurrency model, based on that of SR. JR provides dynamic remote virtual machine creation, dynamic remote object creation, remote method invocation, asynchronous communication, rendezvous, and dynamic process creation ...

Keywords: Concurrency, Java, SR, concurrent object-oriented programming

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Groups Home | Help | Sign in

Google Groups BETA

Web Images **Groups** News Froogle Local **more »**

inner class and template Search Advanced Groups Search Preferences

The **"AND"** operator is unnecessary -- we include all search terms by default. [details]

Members: Sign in
New users: Join

Google Groups

Groups Alerts

Create a new group

About Google Groups

Searched all groups Results **1 - 10** of 4,200 for **inner class and template**

Sorted by relevance Sort by date

inner class in **template class**
... don't have to know this, than **inner class** not equal with a non-**inner class** (for
a normal **class** the above works properly) ? what does the "**template** from the ...
comp.lang.c++.moderated - Aug 6 1999, 3:01 pm by Levente Farkas - 3 messages - 2 authors

Sponsored Links

1000's of Free Templates
Buy 1 **Template** at 15.95 **and** you can have our entire collection for Free
www.onlytemplates.info

Website Templates
Get Unlimited Access To 1000's Of Templates & Web Tools for $19.00
www.templatesforfree.info

See your message here...

C++, templates, CW, **inner classes**...
... A, B>::~TMyTemplate() { // Function **template** <**class** A, **class** B> UInt32 TMyTemplate<A, B>::Function() { // **Inner class** constructor **template** <**class** A, **class** ...
comp.sys.mac.programmer.codewarrior - Jun 24 1999, 9:55 pm by Bruce Horn - 2 messages - 2 authors

inner class in **template class**
... code or just the compilers are bad ?: (the intresting part are in comment:-)) - include <iostream> using namespace std; **template** <**class** T> **class** A ...
comp.lang.c++ - Aug 3 1999, 9:05 am by Levente Farkas - 1 message - 1 author

Can a **template class** have a nested **template class**?
... **template** <> **class Inner**<true> {}; **template** <> **class Inner**<false> {}; ; No ... specialization
caanot appear at **class**-scope: 14.7 ... namespace of which the **template** is a ...
comp.std.c++ - Aug 14 2001, 7:29 pm by Gabriel Dos Reis - 8 messages - 6 authors

Correct syntax for definitions of **inner classes** of a **template** ...
... working on chpater 13 exercises that deal with templates. ... String **class** that uses
the **template** parameter C ... errors for my defintions of the **inner class** Srep, and ...
comp.lang.c++ - Oct 27 2003, 1:19 am by Oplec - 2 messages - 2 authors

friend **template class** within **template class**
... the outer **class** and propogate the **template** parameter(s) to the **inner class** without
actually templating the **inner class** and use the **template** parameter in ...
microsoft.public.vc.language - Mar 2 2000, 4:37 pm by neutrino_sun...@my-deja.com - 2 messages - 2 authors

c++ **template - inner class**
... please help // Upper.h **template** <**class** o> **class** Upper{ private: **class Inner** { private:
Object o; Node* pointer; public: o getO(); **Inner*** getPointer(); ; **Inner*** i ...
microsoft.public.vc.language - Sep 25, 8:56 am by peter - 5 messages - 4 authors

Template inner-class compile/link problem
... reply to my original post: Member templates have to ... use the **template** parameter of
the **template class** within which ... That is to say, my **inner class** behaves as I ...

microsoft.public.vc.language - Aug 19 2002, 2:09 pm by John Barber - 5 messages - 4 authors

Templates/**Inner Classes**
... IC ic; // Instance of **inner class** IC ; **template** <**class** T> T Array<T>::get(int index) {
return data[index]; } **template** <**class** T> T Array<T>::IC::get ...
borland.public.cppbuilder.language - Apr 14 2001, 4:02 pm by Daryl Trumbo - 2 messages - 2 authors

Syntax for member of nested **class template**
... of g++ that I have doesn't understand templates, so I ... of a pair of nested **classes**
template<**class** T> **class** Outer { **class Inner** { **Inner** operator= (**Inner** ...
comp.std.c++ - Sep 18 1993, 2:53 pm by David Arnstein - 2 messages - 2 authors

New! Get the latest messages on **inner class and template** emailed to you with Google Alerts.

Gooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 Next